Document Search using Langchain

Purpose: This Python project searches a PDF document for a given query and retrieves the most relevant passage.

Libraries Used:

- **langchain_community.document_loaders:** This library provides functions to load documents in various formats, including PDFs.
- **transformers**: This library provides pre-trained models for natural language processing (NLP) tasks.
- **sentence-transformers:** This library provides pre-trained models for encoding sentences into vectors.
- **huggingface_hub**: This library allows you to access and use models from the Hugging Face Hub.
- **langchain_community.embeddings**: This library provides classes for embedding text using NLP models.
- **FAISS**: This library implements a fast similarity search algorithm.

Code Breakdown:

1. **Import Libraries:** The code starts by importing the necessary libraries.
2. **Load PDF Document:**
   - **PyPDFLoader** is used to load the PDF document named "**pd.pdf**".
   - The **load_and_split** function splits the document into separate pages.
3. **Load Sentence Embedding Model:**
   - **HuggingFaceEmbeddings** class is used to load a pre-trained sentence embedding model from Hugging Face Hub.
   - The specified model name in this case is **"sentence-transformers/all-MiniLM-L6-v2"**. This model encodes sentences into vectors that capture their semantic meaning.
4. **Embed Query Text:**
   - The text "**Conductor**" (**a sample query**) is converted into a vector using the loaded sentence embedding model.
5. **Create FAISS Index:**

    - FAISS (Fast Approximate Nearest Neighbor Search) is used to create an index for efficient similarity search.
    - The index is built from the document pages (after encoding them into vectors using the sentence embedding model).
6. **Search for Similar Documents:**
    - The **similarity_search** function of the FAISS index is used to find the document page most similar to the query "Conductor".
    - The function returns a list of similar documents, with the most relevant one at the top (**k=1 specifies retrieving only the top result)**.
7. **Print Results:**
    - The code iterates through the returned documents and prints the page number and the first 300 characters of the page content.

Project Usage:

1. Save the code as a Python script (e.g., **document_search.py**).
2. Ensure you have the required libraries installed (p**ip install langchain_community transformers sentence-transformers huggingface_hub faiss**).
3. Replace "**pd.pdf**" with the actual path to your PDF document.
4. Run the script using Python (**python document_search.py**).

This will output the page number and the beginning of the passage in the document that is most relevant to the query "Conductor".

Note:

- This is a basic example. You can modify the code to search for different queries, adjust the number of returned results (k value in **similarity_search**), and potentially explore other document loaders and embedding models depending on your specific needs.